Gestion adaptative de l'énergie pour les infrastructures de type grappe ou nuage

Daniel Balouek-Thomert, Eddy Caron, Laurent Lefèvre

ComPAS'2014 : Parallélisme / Architecture / Système
Neuchâtel, Suisse, du 23 au 25 avril 2014

Gestion adaptative de l'énergie pour les infrastructures de type grappe ou nuage

Daniel Balouek-Thomert[1,2], Eddy Caron[1], and Laurent Lefèvre[1]

[1]LIP, UMR CNRS - ENS de Lyon - INRIA - UCB Lyon 5668, Université de Lyon
[2]NewGeneration SR, 231 rue Saint-Honoré, Paris
{daniel.balouek-thomert,eddy.caron,laurent.lefevre}@ens-lyon.fr

Résumé
Dans un contexte d'utilisation de ressources hétérogènes, la performance reste le critère traditionnel pour la planification de capacité. Mais, de nos jours, tenir compte de la variable énergétique est devenu une nécessité. Cet article s'attaque au problème de l'efficacité énergétique pour la répartition de charge dans les systèmes distribués. Nous proposons une gestion efficace en énergie des ressources par l'ajout de fonctionnalités de gestion des événements liés à l'énergie, selon des règles définies par l'utilisateur. Nous implémentons ces fonctionnalités au sein de l'intergiciel DIET, qui permet de gérer la répartition de charge afin de mettre en évidence le coût des compromis entre la performance et la consommation d'énergie.
Notre solution et son intérêt sont validés au travers d'expériences en évaluant la performance et la consommation électrique mettant en concurrence trois politiques d'ordonnancement. Nous mettons en avant le gain obtenu en terme énergétique tout en essayant de minimiser les écarts de performance. Nous offrons également à l'intergiciel responsable de l'ordonnancement une réactivité face aux variations énergétiques.
Mots-clés : Systèmes distribués, efficacité énergétique, répartition de charge, planification de capacité

1. Introduction

Ces dernières années, le calcul distribué est devenu incontournable dans le domaine des Technologies de l'Information et de la Communication (TIC). La diversité et le nombre de services évoluent de façon constante, tandis que de nouveaux usages ne cessent d'émerger. Le calcul a considérablement évolué, allant de petites machines isolées vers la construction de grandes architectures matérielles, motivée par des besoins grandissants d'efficacité et de dimensionnement. Un cas d'utilisation de ces architectures est le calcul dans les nuages (*Cloud Computing*), qui désigne l'utilisation de serveurs virtualisés distants en tant que service. Ces innovations sont le fruit d'une véritable course à la performance, symbolisées par le TOP500, classement bi-annuel des machines les plus rapides au monde. Ces 20 dernières années, la première machine du classement décrit une augmentation de performance selon un facteur 100 [1]. À l'heure

1. Le classement du TOP500. http ://www.top500.org

actuelle, la perspective de l'exascale est limitée par un facteur majeur : la consommation électrique [9]. Les TIC ont pris une portion importante de la consommation électrique mondiale, et la tendance est à la hausse pour les prochaines années [7, 9]. La liste du Green500 [2] sensibilise à l'efficacité énergétique des supercalculateurs en éditant une liste selon un critère de performance par watt. Ce papier aborde la mise en place de méthodes efficaces en énergie afin de mettre en œuvre de nouvelles stratégies d'ordonnancement, sans impacter les performances des applications et du système.
La validation de notre approche bénéficie de la planification de capacité des ressources par l'intermédiaire du gestionnaire de ressources DIET [2].
Le contenu de cet article est structuré de la manière suivante. La Section 2 présente les travaux antérieurs liés à cette problématique. La Section 3 présente la plateforme expérimentale et ses caractéristiques. La Section 4 évoque l'implémentation de notre cas d'utilisation en exécutant des tâches indépendantes sur un ensemble de machines hétérogènes. La Section 5 détaille notre proposition de planification de capacité réactive au contexte d'exécution. Ces algorithmes sont validés expérimentalement dans la Section 6. La Section 7 conclut ce papier et présente les perspectives.

2. État de l'art

Outre l'intérêt de l'économie d'énergie réduisant la facture des centres de traitement de données en terme de $\frac{\text{nombre de calculs}}{\text{consommation énergétique}}$, on constate que malgré une demande croissante dans le domaine de l'informatique dans les nuages, ces centres sont rarement pleinement utilisés [12]. Les besoins fluctuants avec une faible utilisation en moyenne sont une limitation à l'efficacité énergétique, particulièrement sur les nœuds de calcul avec une faible charge de travail. Les techniques d'économie d'énergie peuvent être réparties selon deux approches : la première consiste à éteindre les ressources matérielles inutilisées alors que la seconde consiste à ralentir les composants matériels des nœuds de calcul [3]. Cependant, dans [14], Le Sueur *et al.* ont mis en évidence le fait que ces techniques sont devenues moins pertinentes sur les machines modernes.
Les logiciels actuels opèrent en passant l'ensemble du serveur dans des modes de veille [1, 8]. Ces améliorations sont appropriées à l'informatique dans les nuages, qui consiste à virtualiser des infrastructures pour traiter tous types de requêtes.
D'un autre côté, les grilles et les infrastructures de calcul opportunistes ont pour but de "mettre en place le partage et la coordination des ressources pour la résolution des problèmes au sein d'infrastructures virtuelles, dynamiques et pluri-institutionelles" [10]. Dans ce contexte, il est plus difficile de mettre en veille ou d'éteindre des serveurs pour des raisons de disponibilité des services.
Beaucoup de travaux supposent que des nœuds d'un cluster homogène ont la même consommation énergétique [11, 13]. En pratique, dû à leurs différentes utilisations, les nœuds d'un même cluster présentent des performances et des consommations différentes. Il a été montré [16] que ces différences peuvent être causées par des facteurs externes, tel que la température extérieure ou bien la position du nœud dans la baie. En complément, Diouri *et al.* [6] ont observé que cette hétérogénéité de consommation pouvait être due à un usage spécifique de composants matériels et de perte d'énergie variant au cours du temps.
Ces études suggèrent le design et l'implémentation de politique d'ordonnancement spécifique aux applications et aux infrastructures matérielles cibles.

2. Le classement Green500. http ://www.green500.org

3. Plateforme Expérimentale

3.1. L'intergiciel

Le projet open source DIET [2] porte ses efforts sur l'implémentation d'un intergiciel dimensionnant (permettant le passage à l'échelle), en se basant sur une architecture hiérarchique visant à distribuer le problème de l'ordonnancement sur plusieurs agents. L'implémentation utilise la technologie CORBA, et par conséquent, bénéficie des nombreux services stables et standardisés fournis par les implémentations libres d'accès de CORBA.

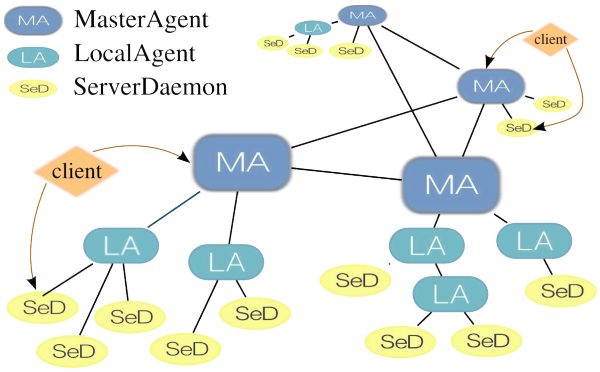


FIGURE 1 – Une hiérarchie DIET et ses différents agents.

La boîte à outils DIET est composée de plusieurs éléments, illustrés sur la Figure 1. Le premier élément est le client, une application utilisant l'infrastructure DIET pour résoudre des problèmes de façon distante. Le second élément est le **SED** (**Server Daemon**) qui agit en tant que fournisseur de service, en exposant des fonctionnalités à travers une interface standardisée ; un SED peut offrir un ou plusieurs services de calcul. Le troisième élément de l'architecture DIET est l'**agent**. Déployé seul ou bien au sein d'une hiérarchie, l'agent facilite la localisation des services et les interactions entre clients et SeDs. La hiérarchie d'agents fournit des services de haut niveau tels que l'ordonnancement et la gestion des données. Le sommet de la hiérarchie d'agents est appelée le **Master Agent (MA)** tandis que les autres sont les **Local Agents(LA)**.
Les applications présentes sur la plateforme DIET sont en mesure d'exercer un certain contrôle sur le sous-système d'ordonnancement par l'intermédiaire de modules spécifiques, appelés plug-in schedulers. Ces fonctionnalités sont invoquées après qu'un utilisateur ait soumis une requête au Master Agent, qui se charge de transmettre la requête aux autres agents de la hiérarchie.
Par exemple, un SED qui fournit un service interagissant avec des bases de données peut bénéficier de l'inclusion d'information sur les bases de données présentes dans le cache du disque, afin de choisir le serveur le plus approprié à chaque requête cliente. Le développeur a la possibilité d'implémenter et d'inclure une fonction personnalisée d'estimation de la performance au sein de chaque SED puis DIET associera la fonction personnalisée d'estimation avec le service. Ainsi, chaque fois qu'une requête client est reçue par le SED en question, un vecteur d'estimation de performance sera transmis à l'émetteur de la requête.

3.2. La plate-forme GRID'5000

La plateforme de test GRID'5000 a été conçue dans le but de supporter la recherche expérimentale sur les systèmes parallèles et distribués. L'utilisation de GRID'5000 permet aux utilisateurs d'effectuer des expériences à tous les niveaux de la pile logicielle. Située en France, GRID'5000 est composée de 26 clusters hétérogènes pour un total de 1100 nœuds répartis sur 9 sites physiques interconnectés par un réseau dédié 10 Gbps.
Afin de mettre en place notre infrastructure DIET, nous utilisons 14 nœuds physiques du site de Lyon dont les spécifications sont présentées dans la Table 1. Les nœuds sont connectés par l'intermédiaire d'un switch avec une bande passante de 1Gbit/s. Le système d'exploitation Debian Wheezy est utilisé sur chacun des nœuds.

Cluster	Nœuds	CPU	Mémoire	Rôle
Orion	4	2x6cœurs @2.30Ghz	32GB	SED
Sagittaire	4	2x1cœurs @2.40Ghz	2GB	SED
Taurus	4	2x6cœurs @2.30Ghz	32GB	SED
Sagittaire	1	2x1cœurs @2.40Ghz	2GB	MA
Sagittaire	1	2x1cœurs @2.40Ghz	2GB	Client

TABLE 1 – Caractéristiques des nœuds physiques.

La surveillance des nœuds est effectuée à l'aide de wattmètres externes du type Omegawatt. Cette infrastructure de supervision de l'énergie, également utilisé dans [5], permet d'obtenir chaque seconde la moyenne de la consommation énergétique en Watts basée sur près de 4000 échantillons par machine [4].

4. Ordonnancement énergétique personnalisé

Comme nous l'avons précisé en Section 3.1, l'intergiciel DIET offre des mécanismes permettant de réaliser nos propres placements des tâches de calcul selon des critères prédéfinis. Dans le contexte de nos travaux, nous avons donc implémenté 3 plug-ins d'ordonnancement DIET basés sur les critères suivants :

– Performance du noeud (La plus grande valeur étant la meilleure). On va donc ici privilégier les nœuds les plus performants. Cette politique est nommée PERFORMANCE.
– Consommation énergétique du noeud (La plus petite valeur étant la meilleure).On va donc privilégier les nœuds les plus efficaces en énergie. Cette politique est nommée ENERGY.
– Placement aléatoire. Cette politique est nommée RANDOM.

Le critère de performance est basé sur une mesure de la performance du noeud lorsque l'ensemble des cores est utilisé. Il en résulte une valeur en flops, indiquant le nombre d'opérations en virgule flottante à la seconde. Ces mesures sont effectuées en utilisant les outils ATLAS [3], HPL [4] et Open MPI [5].

3. Automatically Tuned Linear Algebra Software. http ://sourceforge.net/projects/math-atlas/
4. Implémentation portable du banc d'essai Linpack pour les architectures de calcul haute performance. http ://www.netlib.org/benchmark/hpl/
5. Implémentation haute performance de la librairie MPI (Open Source). http ://www.open-mpi.org/

La métrique de consommation électrique est basée sur le nombre de requêtes précédemment traitées par un SED pondéré par la mesure de la consommation de la machine depuis le démarrage du serveur. Par conséquent, chaque fois qu'un client soumet une requête, chacun des SED récupère sa propre consommation électrique et le nombre total de requêtes.
D'autres critères existent dans la littérature, impliquant la prise en compte de la consommation énergétique de la machine machine physique lorsqu'elle n'est pas en activité (*idle*) [6] ou bien son taux d'utilisation [15].

5. Provisionnement de serveurs avec considération de la variable énergétique

Dans la section précédente, nous avons mis en place un placement des tâches personnalisé selon des critères définis au niveau de l'ordonnanceur. Un autre aspect de notre contribution est d'adapter le provisionnement de capacité des ressources en prenant en compte des événements liés à la gestion de l'énergie, tels que les fluctuations du prix de l'électricité ou les pics de chaleur.
Cette fonctionnalité permet à l'ordonnanceur d'effectuer des opérations de façon automatique et autonome en vérifiant des seuils d'exploitation avant d'exécuter des décisions de placement ou de provisionnement. Dans cet exemple, nous considérons que l'administrateur définit des seuils relatifs aux variables suivantes :
– Coût de l'énergie pour une période donnée
– Conditions locales de température
Ces informations sont généralement disponibles à l'aide de prédictions basées sur des données antérieures, par l'intermédiaire d'horaires fournis par le fournisseur d'énergie ou bien en utilisant les systèmes internes de monitoring de l'architecture.
En utilisant ces variables, le Master Agent peut définir une prévision de l'utilisation des ressources en adaptant le nombre de nœuds disponibles pour le calcul. Nous utilisons le terme de *nœuds candidats* pour se référer à ces machines. Le processus de décision correspond aux étapes suivantes :

1. *Soumission d'un problème*

   Un client soumet une requête décrivant le problème et le service souhaité. Si aucun des SEDs n'est en mesure de le résoudre, une erreur est retournée au client.

2. *Propagation de la requête*

   Le Master Agent communique avec l'ensemble de ses agents dans le but de transmettre la requête au SEDs.

3. *Collecte des métriques d'estimations*

   Chaque SED récupère ses métriques d'estimations personnalisées, en particulier celles associées à la performance et à la consommation d'énergie. Une réponse contenant le vecteur d'estimation est retournée au Master Agent.

4. *Vérification des seuils*

   Le Master Agent vérifie les seuils de température et de coût de l'énergie afin d'ajuster le nombre de nœuds candidats.

5. *Tri des candidats*

   Une fois que le Master Agent récupère l'ensemble des vecteurs d'estimation, il effectue un tri en fonction du critère spécifié (dans le cas d'un tri aléatoire, il attribue un nombre aléatoire à chacun des SED). Le premier SED est élu et notifié.

6. *Résolution du problème*

   Le client contacte le SED élu, qui peut démarrer la résolution du problème soumis.

7. *Retour du nombre de candidats au client*

   Une notification contenant le nombre actuel de nœuds candidats est transmis aux agents clients, permettant ainsi un ajustement du flux de requêtes.

Dans la Section 6, nous illustrons ces mécanismes par l'intermédiaire de deux scénarios d'expérimentations nécessitant l'exécution d'un ensemble de tâches indépendantes.

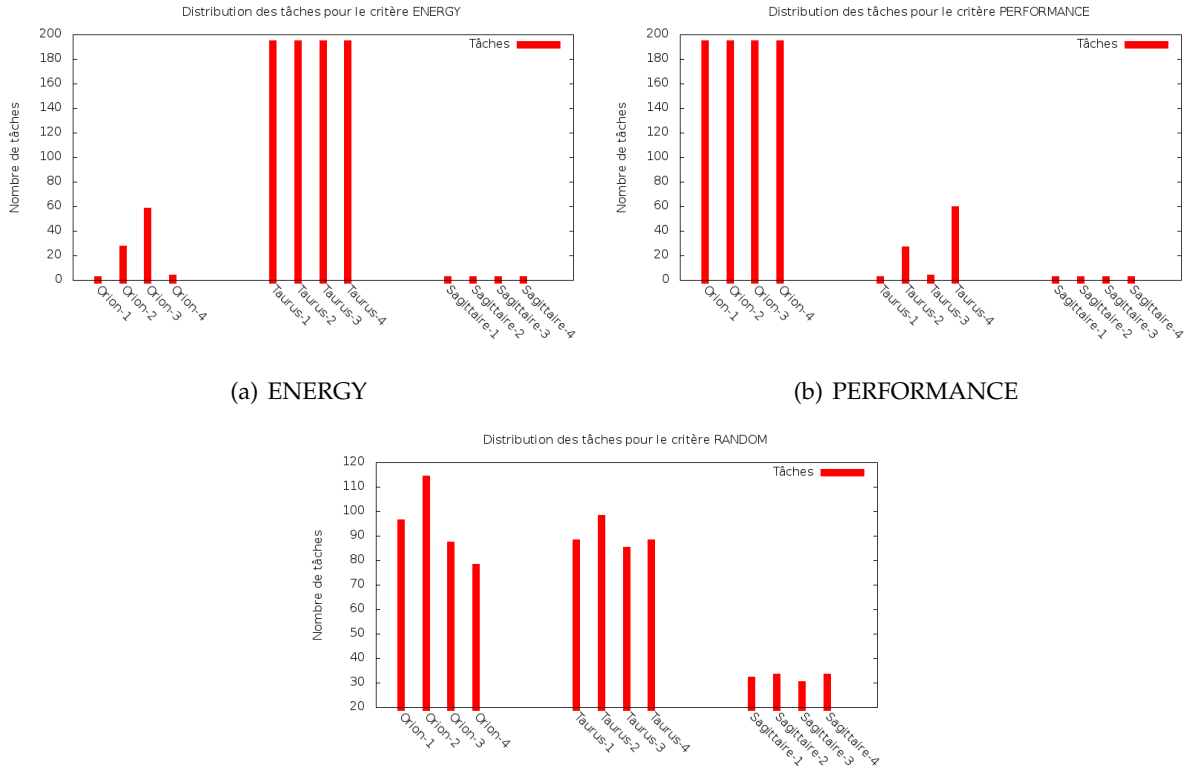
6. Resultats

6.1. Répartition de la charge de la travail

La première expérience a pour but de comparer la distribution des tâches parmi les nœuds de calcul, selon les différentes politiques d'ordonnancement que nous avons présentés Section 4.
Soit n le nombre total de cores présents au sein de l'infrastructure DIET. Nous considérons un client qui soumet un ensemble de $10n$ requêtes. Le problème à résoudre est une application de calcul intensif, consistant en 10^7 additions successives sur une variable. Tous les serveurs sont capables de résoudre le problème, et chaque nœud ne peut exécuter plus d'une tâche par cœur de façon simultanée.
La distribution temporelle des requêtes contient deux phases :
– Première phase : Soumission simultanée de n requêtes par le client
– Seconde phase : Soumission de 2 requêtes par seconde
En considérant que l'ordonnanceur ne possède aucune information préalable sur les nœuds, les premières requêtes permet au Master Agent d'obtenir un profil d'exécution dynamique des machines.
La Figure 2 présente les résultats de cette expérience en mettant en évidence le nombre de tâches traitées par chacun des nœuds.
La répartition des tâches en utilisant la consommation énergétique (Figure 2 (a)) comme critère de placement montrent que les nœuds *Taurus* sont les plus efficaces énergétiquement, exécutant la majorité des tâches. L'équilibre de charge est similaire sur la (Figure 2 (b)), lorsque la performance est privilégiée. Cependant, la majorité des tâches est exécutée par les nœuds *Orion*. La répartition aléatoire des tâches (Figure 2 (c)) montrent que les nœuds *Sagittaire* exécutent moins de tâches que les autres clusters. Ceci est dû au fait qu'une tâche est exécutée plus lentement sur ces nœuds. Par conséquent, ils sont moins disponibles lorsque les décisions d'ordonnancement sont effectuées.
La Figure 3 présente la consommation de l'ensemble de l'architecture, groupée par clusters. Il est utile de préciser que la consommation énergétique des agents DIET est constante durant le test des différents critères. Par conséquent, elle ne présente pas d'influence sur les résultats.

	RANDOM	ENERGY	PERFORMANCE
Temps d'exécution (s)	2336	2321	2228
Consommation énergétique (J)	6041436	4528547	5618175

TABLE 2 – Comparaison du temps d'exécution et de la consommation énergétique entre les politiques d'ordonnancement



(a) ENERGY (b) PERFORMANCE

(c) RANDOM

FIGURE 2 – Répartition des tâches selon les différents critères. (a) Consommation énergétique. (b) Performance. (c) Aléatoire.

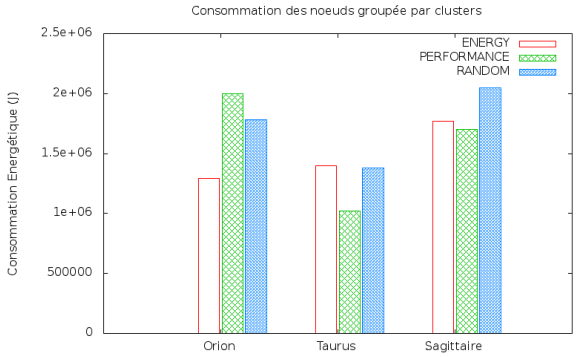


FIGURE 3 – Consommation énergétique par cluster.

Nous utilisons le tableau 2 pour comparer les durées d'exécution et la consommation énergétique en fonction du critère d'ordonnancement choisi. Le meilleur cas est observé lorsque le critère privilégié est un nombre élevé de Flops (PERFORMANCE). En comparant cette valeur avec la durée d'exécution du critère ENERGY, nous avons observé une perte de performance de l'ordre de 6%.
En terme de consommation énergétique, le critère ENERGY présente un gain de 25% lorsqu'il est comparé à RANDOM, et jusqu'à 19% en comparaison avec PERFORMANCE.

La distribution aléatoire apparaît comme le pire cas, puisqu'elle garantit que toutes les ressources sont amenées à être utilisées durant l'expérience, avec pour effet une consommation énergétique plus élevée. L'usage de nœuds plus lents impacte également la performance mais cette tendance est atténuée par un effet de recouvrement (les nœuds les plus rapides exécutant plus de tâches en parallèle).

6.2. Provisionnement de ressource par considération du contexte d'exécution

La seconde expérience a pour but de démontrer la réactivité de l'ordonnanceur en considérant les variations de deux métriques : le coût de l'énergie et la température.

Concernant l'énergie, nous définissons 3 valeurs : 1.0 pour le coût le plus élevé, 0.8 en tant que coût intermédiaire et 0.5 comme étant le coût le plus bas. Dans un but de simplification, nous définissons le coût de l'énergie comme un ratio entre le coût théorique maximum et le coût effectif pour une période donnée.

La métrique de température considère deux seuils : une plage d'utilisation normale pour une mesure de température inférieure à 25° et une plage d'utilisation non tolérée si la température est supérieure à 25°.

Nous décrivons ensuite 4 événements, répartis selon deux catégories. D'abord, les événements planifiés correspondants à une prédiction ou une annonce prédéfinie, qui peut être détectée avant son occurrence par l'ordonnanceur. Ensuite, les événements imprévus pouvant être détectés seulement au moment précis de l'occurrence.

Soit t le temps de départ de l'expérience, nous définissons :

Événement 1 À l'instant $t_0 + 60$ min, le coût de l'énergie diminue à 0.8 et descend sous un premier seuil (événement planifié)

Événement 2 À l'instant $t_0 + 120$ min, le coût de l'énergie chute à 0.5 et descend sous un second seuil (événement planifié)

Événement 3 À l'instant $t_0 + 160$ min, une hausse de la température est détectée (événement imprévu)

Événement 4 À l'instant $t_0 + 240$ min, la température retrouve une valeur tolérée (événement imprévu)

Le statut de l'infrastructure correspond donc à la valeur des différentes métriques à un instant *t*. Dans cet exemple, nous considérons que le Master Agent consulte le statut de la plateforme toutes les 10 minutes, avec la possibilité d'être informé des évenements planifiés à un intervalle de temps $t + 2$.

Sur la Figure 4, nous présentons un échantillon du fichier XML décrivant le statut du serveur. Un échantillon correspond à un instant spécifique. Pour chaque échantillon, nous définissons trois *tags*, à savoir `<température>`, `<candidates>`, `<energy_cost>`. À chaque intervalle de temps, le Master Agent lit les valeurs et prend les décisions associées. Par conséquent, les informations futures, telles que les prévisions, peuvent être ajoutées au fichier de statut pour permettre la prise de décisions anticipée. Les tags et les intervalles de temps sont des données variables pouvant être ajustées en fonctions des contextes.

Il est nécessaire de définir les seuils et les actions associées aux franchissements. Les actions peuvent être définies par l'intermédiaire de scripts externes ou au sein de l'ordonnanceur. Dans cet exemple, nous avons implémenté quatre comportements associés aux métriques de l'expérience. Soit *c* le coût de l'énergie pour une période donnée et T la température mesurée à un instant t.

– si $T > 25°$ alors candidates_nodes $=$ 20% des nœuds disponibles

```
<timestamp value="1385896446">
  <temperature>23.5</temperature>
  <candidates>8</candidates>
  <energy_cost>0.6</energy_cost>
</timestamp>
```

FIGURE 4 – Échantillion du fichier de statut du serveur.

- si 1.0 >= c > 0.8 alors candidates_nodes = 40% des nœuds disponibles
- si 0.8 >= c > 0.5 alors candidates_nodes = 70% des nœuds disponibles
- si c >= 0.5 alors candidates_nodes = 100% des nœuds disponibles

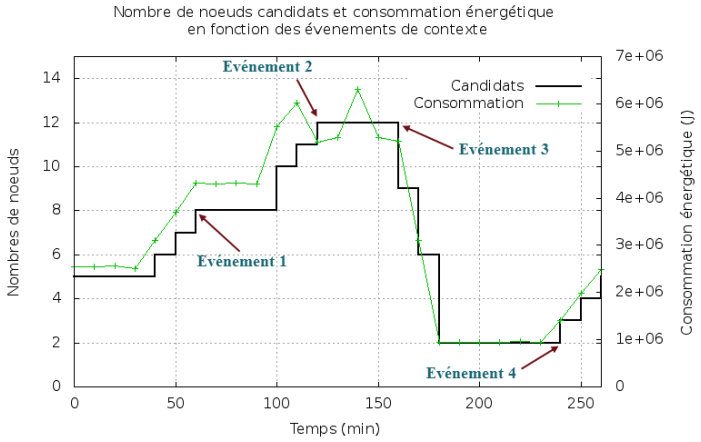


FIGURE 5 – Comparaison entre le nombre de nœuds candidats et la consommation énergétique

La Figure 5 présente l'évolution du nombre de candidats et la consommation électrique en fonction des événements.(Les croix désignent la valeur moyenne des mesures d'énergie durant les 10 minutes précédentes.)
La configuration matérielle des nœuds gérés par l'intergiciel est identique à l'expérience précédente (cf. Tableau 1). L'expérience commence avec un coût de l'énergie d'une valeur de 1.0. L'**événement 1** est une baisse du coût de l'énergie à $t_0 + 60$ min. Le Master Agent prévoit donc de mettre à disposition 8 nœuds candidats à $t_0 + 60$ min. Le lot de machines candidates est incrémentée par vagues afin d'éviter les effets de bord liés à la mise en service simultanée de machines. Nous pouvons observer une augmentation linéaire de la consommation énergétique au sein de l'architecture.
L'**événement 2** est traité de façon similaire à l'**événement 1**. Le coût de l'énergie permet l'utilisation de tous les nœuds de l'architecture. Ces nœuds sont ajoutés au lot de nœuds candidats durant les 20 minutes suivant la prise de connaissance de l'information, résultant de l'usage de 12 nœuds entre $t_0 + 120$ et $t_0 + 160$ min.
L'**événement 3** simule une augmentation soudaine de la température, détectée par le Master Agent à l'instant $t_0 + 160$ min. Dans le contexte de cette expérience, les règles administrateurs préconisent la réduction du nombre de nœuds candidats au nombre de 2. Cette opération est effectuée en 3 étapes, dans le but de provoquer une chute de la température (simulée) et de

l'énergie (effective et mesurée).
Le système continue d'opérer avec deux nœuds candidats jusqu'à ce qu'une température tolérée soit mesurée à l'instant t_0+240 min (**Évènement 4**). Le Master Agent prend alors la décision d'augmenter le nombre de machines candidates toutes les 10 minutes, jusqu'à atteindre à nouveau la valeur maximale de 12.
Ce scénario expérimental met en évidence la réactivité de l'ordonnanceur face à des événements de type différents. La fréquence des points d'observation, ainsi que les réactions correctives associées sont entièrement paramétrables et permettent l'utilisation d'outils tiers pour la prédiction d'événements et l'ajustement des seuils d'exploitation. Ainsi, l'exploitant dispose de mécanismes permettant de classer les machines selon un critère d'efficacité énergétique et de borner leur consommation énergétique globale au cours du temps.

7. Conclusions et Perspectives

Il est maintenant crucial de concilier les besoins en performance tout en réduisant la consommation énergétique au sein des architectures de calculs distribuées.
Nous avons exploité l'hétérogénéité de l'infrastructure, en utilisant un ensemble de métriques pour établir un profil d'exploitation dynamique des éléments de calcul. Ces données sont collectées et traitées de façon personnalisée par l'intermédiaire de l'intergiciel DIET.
La première partie présente la comparaison de politiques d'ordonnancement basées sur la performance et la consommation énergétique. Les résultats montrent une réduction significative de la consommation en énergie de l'ordre de 25%, tandis que la performance présente une perte mineure (6% de différence en comparant les durées d'exécution).
La seconde partie de notre proposition décrit un provisionnement automatique de capacité des ressources, en fonction de critères externes, tels que le coût de l'énergie ou les fluctuations de température au sein des ressources. Les résultats expérimentaux montrent que l'ordonnanceur peut se montrer réactif aux événements en appliquant des comportements de réaction définis par l'administrateur, permettant un contrôle de la consommation énergétique globale ou la gestion d'un budget énergétique.
Nos travaux futurs visent à raffiner l'ordonnancement en ajoutant des critères de localité spatiale dans la perspective de corréler les échanges thermiques et l'ordonnancement au sein des centres de calculs.

Remerciements

Les évaluations expérimentales décrites dans cet article ont été en partie conduites sur l'environnement GRID'5000, projet soutenu par l'action de développement Inria ALADDIN avec un soutien du CNRS, RENATER et différentes Universités [6].

Bibliographie

1. Beloglazov (A.) et Buyya (R.). – Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Trans. Parallel Distrib. Syst*, vol. 24, n7, 2013, pp. 1366–1379.
2. Caron (E.) et Desprez (F.). – DIET : A scalable toolbox to build network enabled servers on the grid. *International Journal of High Performance Computing Applications*, vol. 20, n3, 2006, pp. 335–352.

6. The GRID'5000 testbed. http ://www.grid5000.fr

3. Carrera (E. V.), Pinheiro (E.) et Bianchini (R.). – Conserving disk energy in network servers. – In *Proceedings of the 2003 International Conference on Supercomputing (ICS-03)*, pp. 86–97, New York, juin 23–26 2003. ACM Press.
4. De Assuncao (M. D.), Gelas (J.-P.), Lefèvre (L.) et Orgerie (A.-C.). – The green grid'5000 : Instrumenting and using a grid with energy sensors. *In : Remote Instrumentation for eScience and Related Aspects*, pp. 25–42. – Springer, 2012.
5. De Assuncao (M. D.), Orgerie (A.-C.) et Lefèvre (L.). – An analysis of power consumption logs from a monitored grid site. – In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pp. 61–68. IEEE, 2010.
6. Diouri (M. E. M.), Glück (O.), Lefèvre (L.) et Mignot (J.-C.). – Your cluster is not power homogeneous : Take care when designing green schedulers ! 2013.
7. Dongarra (J.), Beckman (P.), Moore (T.), Aerts (P.), Aloisio (G.), Andre (J.-C.), Barkai (D.), Berthou (J.-Y.), Boku (T.), Braunschweig (B.), Cappello (F.), Chapman (B.), Chi (X.), Choudhary (A.), Dosanjh (S.), Dunning (T.), Fiore (S.), Geist (A.), Gropp (B.), Harrison (R.), Hereld (M.), Heroux (M.), Hoisie (A.), Hotta (K.), Jin (Z.), Ishikawa (Y.), Johnson (F.), Kale (S.), Kenway (R.), Keyes (D.), Kramer (B.), Labarta (J.), Lichnewsky (A.), Lippert (T.), Lucas (B.), Maccabe (B.), Matsuoka (S.), Messina (P.), Michielse (P.), Mohr (B.), Mueller (M. S.), Nagel (W. E.), Nakashima (H.), Papka (M. E.), Reed (D.), Sato (M.), Seidel (E.), Shalf (J.), Skinner (D.), Snir (M.), Sterling (T.), Stevens (R.), Streitz (F.), Sugar (B.), Sumimoto (S.), Tang (W.), Taylor (J.), Thakur (R.), Trefethen (A.), Valero (M.), van der Steen (A.), Vetter (J.), Williams (P.), Wisniewski (R.) et Yelick (K.). – The International Exascale Software Project roadmap. *The International Journal of High Performance Computing Applications*, vol. 25, n1, février 2011, pp. 3–60.
8. Feller (E.), Rilling (L.) et Morin (C.). – Snooze : A scalable and autonomic virtual machine management framework for private clouds. 2012, pp. 482–489.
9. Feng (W.-C.), Feng (X.) et Ge (R.). – Green supercomputing comes of age. *IT Professional*, vol. 10, n1, 2008, pp. 17–23.
10. Foster (I.), Zhao (Y.), Raicu (I.) et Lu (S.). – Cloud computing and grid computing 360-degree compared. *CoRR*, vol. abs/0901.0131, 2009.
11. Ge (R.), Feng (X.), Song (S.), Chang (H.-C.), Li (D.) et Cameron (K. W.). – PowerPack : Energy profiling and analysis of high-performance systems and applications. *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, n5, mai 2010, pp. 658–671.
12. Hawkins (A.). – Unused servers survey results analysis. *The Green Grid*, 2010.
13. Kim (K. H.), Buyya (R.) et Kim (J.). – Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. 2007, pp. 541–548.
14. Le Sueur (E.) et Heiser (G.). – Dynamic voltage and frequency scaling : The laws of diminishing returns. – In *Proceedings of the 2010 international conference on Power aware computing and systems*, pp. 1–8. USENIX Association, 2010.
15. Lim (S.-H.), Sharma (B.), Tak (B.-C.) et Das (C. R.). – A dynamic energy management scheme for multi-tier data centers. 2011, pp. 257–266.
16. Varsamopoulos (G.), Banerjee (A.) et Gupta (S. K. S.). – Energy efficiency of thermal-aware job scheduling algorithms under various cooling models. – In Ranka (S.), Aluru (S.), Buyya (R.), Chung (Y.-C.), Dua (S.), Grama (A.), Gupta (S. K. S.), Kumar (R.) et Phoha (V. V.) (édité par), *Contemporary Computing - Second International Conference, IC3 2009, Noida, India, August 17-19, 2009. Proceedings*, *Communications in Computer and Information Science*, volume 40, pp. 568–580. Springer, 2009.